Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 9 (2012) 226 – 235

Procedia Computer Science

International Conference on Computational Science, ICCS 2012

Dimensioning Scientific Computing systems to improve performance of Map-Reduce based applications

Gabriel G. Castañé[a], Alberto Núñez[b], Rosa Filgueira[c], Jesús Carretero[a]

[a]*Computer Architecture Group, Computer Science Department, Universidad Carlos III de Madrid, Spain*
[b]*Departamento de Sistemas Informáticos y Computación, Universidad Complutense de Madrid, Spain*
[c]*School of Informatics, University of Edinburgh, Scotland*

Abstract

Map-Reduce is a programming model widely used for processing large data sets on scientific clusters. Most of the efforts and research are focused on enhancing and alleviating the drawbacks of the model proposed by Google.

The requirements of Map-Reduce based applications are often unclear because of the difficulty in satisfying the overall system throughput, as well as exploring alternatives to obtain a good tradeoff between the performance of basic systems such as storage, networking and CPU. In this paper we present an evaluation of the compared performance of scaling up scientific computing systems using a Map-Reduce application model. This work is specifically focused on medium-size multi-core systems, frequently used by researchers to compute scientific applications. The scaling process is oriented towards the three main resources: computing power, communications and storage.

By performing an extensive set of simulations using iCanCloud simulator, we also show that main bottlenecks of those kinds of applications executed in cluster systems are found in storage and network systems. Thence, in order to increase the overall performance of those applications, the computing power must be scaled up proportionally along the network and storage system.

Keywords:
Map-Reduce applications, scientific applications, modeling and simulation, scientific clusters, performance prediction

1. Introduction

Today, scientists need powerful systems to process large amounts of data in a reasonable time-frame. Map-Reduce applications are causing an explosive growth in the size of scientific data-sets. In fact, Map-Reduce model [1] is widely used by commercial applications, scientific applications and web search engines which require to manage large amounts of data on a daily basis [2][3]. Commonly, these applications implement data mining algorithms using Map-Reduce, where large data-sets are split into small chunks to be processed in cluster nodes, which provides a high level of scalability.

Email addresses: ggonzalez@arcos.inf.uc3m.es (Gabriel G. Castañé), alberto.nunez@pdi.ucm.es (Alberto Núñez), rosa.filgueira@ed.ac.uk (Rosa Filgueira), jcarrete@arcos.inf.uc3m.es (Jesús Carretero)

doi:10.1016/j.procs.2012.04.024

Data intensive applications perform massive I/O operations and require high amounts of computing. However, the CPU and the storage system are not the only concerns in the overall performances of those systems. The network system also plays an important role in performance. However, efficiently exploiting those resources requires using the right architectural configuration, which in most cases is difficult to design.

Commodity clusters have been the main resource for high performance computing (HPC) due to their high cost-effectiveness. As an example, in the most recent survey of the fastest 500 computers in the world, 82.2% are clusters, where a great part of those are medium-size clusters (see the June 2011 Top500 list [4]). Moreover, 44.2% of those systems used Gigabit Ethernet and many of them still use Fast Ethernet (100 Megabits/second).

The biggest advantage of this kind of scientific clusters is the possibility to scale easily by adding more nodes to the system. This method of scaling up the architecture allows improving the performance of some aspects of the system like the number of computing units or the number of storage servers. Nonetheless, there are other aspects of the infrastructure that cannot be scaled up that easily and end up becoming bottlenecks for the overall system performance. Thus, a proper dimensioning of computing power, network interconnection and storage is an essential issue that needs to be resolved in order to improve overall system performance.

At present, high-performance computing has reached a turning point where computing power is no longer the most important concern. However, the focus is shifting from the main optimization only in the computing system, to optimizing other systems, such as the storage and the networking systems. New technologies also incorporate more confusion on the table. The rise of multi-core processors makes it very easy to increase the number of computing units. But it also creates a gap in the communications between those processes that belong to the same node and the rest. While the local processes can intercommunicate very fast, the communication outside the node suffers more saturation because more processes require using the same communication channel at the same time. The latter is also true for accessing the storage servers, which also suffer from a bigger saturation. Hereby, we focus on obtaining optimal performance for Map-Reduce based applications that are both CPU-intensive and I/O intensive, which is currently a challenging problem.

The main goal of this work is to obtain those architectural configurations of scientific clusters that provide the best performance for Map-Reduce based applications. The main difficulty of this process is the high number of inter-related features that has a direct impact on the overall system performance. And this, in turn, hampers the task of making a choice of such architecture. Thus, we have modeled Map-Reduce and three different scientific clusters architectures using the iCanCloud simulation platform [5].

Thence, the novelty of our proposal is to explore several architectural configurations for evaluating a Map-Reduce application model in terms of performance, instead of improving the Map-Reduce model itself. We focus on applications that execute on small to medium sized scientific clusters, which combine the exploitation of CPU parallelism with massive data operations. Therefore, we consider a comparison of the results obtained by using different architectural configurations for each system, whereof we can set the right size, in terms of resources, for each system to obtain the best performance for a given application. This evaluation has been performed using the iCanCloud simulation platform.

The main contributions of this work are:

1. We show guidelines to make an architectural configuration choice that provides the better performance for the Map-Reduce based applications.
2. We present a model for simulating Map-Reduce applications based on the Hadoop implementation. Also, a set of performance tests has been achieved to check the accurateness of our proposed model.
3. We show that our evaluation provides an insight into dimensioning the resources of a specific architecture to obtain the best performance for a given Map-Reduce based application model.

The rest of the paper is structured as follows: in Section 2 some related work is described. Section 3 presents a brief overview of the iCanCloud simulation platform. Section 4 presents a validation of the simulation platform used in this work. Section 5 presents the design of the Map-Reduce application model used in this work. Section 6 presents the model of different scientific cluster architectures where experiments are simulated. Section 7 presents a set of experiments and their corresponding results. Finally, section 8 presents conclusions and suggests some future work.

2. Related work

Currently, simulation techniques are increasingly becoming a central activity in the design of new systems and in the analysis of existing ones, since they enable designers and researchers to investigate systems behavior through virtual representations. In fact, the abundance of existing simulators becomes overwhelming because of the inherent difficulty to select a single simulator for a specific domain for which they were designed.

Some of those simulators are focused on simulating the entire system with a very high level of detail by providing functional execution of unmodified commercial operating systems and applications. Those simulators are called full-system simulators. The main drawback of this type of simulators is the slowdown factor in execution time (e.g. 10.000X), being impractical for the analysis of scientific cluster systems. Some examples are COTSon [6], SimOS [7] [8] and the SST (Structural Simulation Toolkit) [9].

Following set of simulators focuses on simulating grid enviroments and cloud computing systems such as GridSim [10], OptorSim [11], SimGrid [12] and MicroGrid [13], CloudSim [14], MDCSim [15] among others. These tools can simulate brokerage of resources or execution of different types of applications on different types of computing resources, but they lack the details to simulate a full-system with accuracy.

Another approach focuses on using performance analysis tools. The main difference between those tools and simulators is that those tools need the application to be executed on the real system, and simulators let the application to be executed in a virtual environment, which does not require the real system. Some examples of those tools are HPCToolkit [16], Scalasca [17], and MPI-NetSim [18].

The main issue of simulators and tools is that they do not obtain a good trade-off between scalability, flexibility and performance. Moreover, most of them lack of a high detailed storage system models. Instead, the iCanCloud simulation platform has been designed to provide providing a high detailed storage model, and to balance flexibility, scalability, performance and accuracy.

During last years, researchers have put a lot of effort into improving and alleviating the main drawbacks of Map-Reduce. As an example, this work [19] present an efficient hierarchical clustering method of mining large datasets with Map-Reduce. The method includes two optimization techniques: Batch Updating to reduce the computational time and communication costs among cluster nodes, and Co-occurrence based feature selection to decrease the dimension of feature vectors and eliminate noise features.

The authors of [20] propose some methods to alleviate the limitations of the Map-Reduce model for spatial applications. Specifically, those methods consists of: 1) a splitting method for balancing workload, 2) pending file structure and redundant data partition dealing with relation between spatial objects, 3) a strip-based two-direction plane sweeping algorithm for computation accelerating.

This paper [21] proposes two novel algorithms, ADABOOST.PL (Parallel ADABOOST) and LOGITBOOST.PL (Parallel LOGITBOOST), that facilitate simultaneous participation of multiple computing nodes to construct a boosted classifier. Those algorithms have been implemented using the Map-Reduce framework and experimented on a variety of synthetic and real-world data sets to demonstrate the performance in terms of classification accuracy, speed-up and scale-up.

The work described in [22] presents and evaluates two different approaches to cope with the synchronization drawback of existing Map-Reduce implementations. The first approach, hierarchical reduction, starts a reduce task as soon as a predefined number of map tasks completes; then it aggregates the results of different reduce tasks following a tree structure. The second approach, incremental reduction, starts a predefined number of reduce tasks from the beginning and has each reduce task incrementally reduce records collected from map tasks

The authors of [23] focus on predictive modeling of multi-relational data such as dyadic data with associated covariates or “side-information”. They give some illustrative examples of applications that involve such data and then describe a general framework based on Simultaneous CO-clustering And Learning (SCOAL), which applies a divide-and-conquer approach to data analysis. This work also shows that the main elements of the SCOAL algorithm can be effectively parallelized using the Map-Reduce framework. Experiments on Amazon EC2 demonstrate that the proposed parallelization results in considerable improvements in run time when using a cluster of machines.

3. Overview of iCanCloud Simulation platform

The iCanCloud simulation platform is oriented towards the simulation of different kinds of cloud computing systems and their underlying architectures. This project was initiated in 2010 and is currently available at [24] as an open source software.

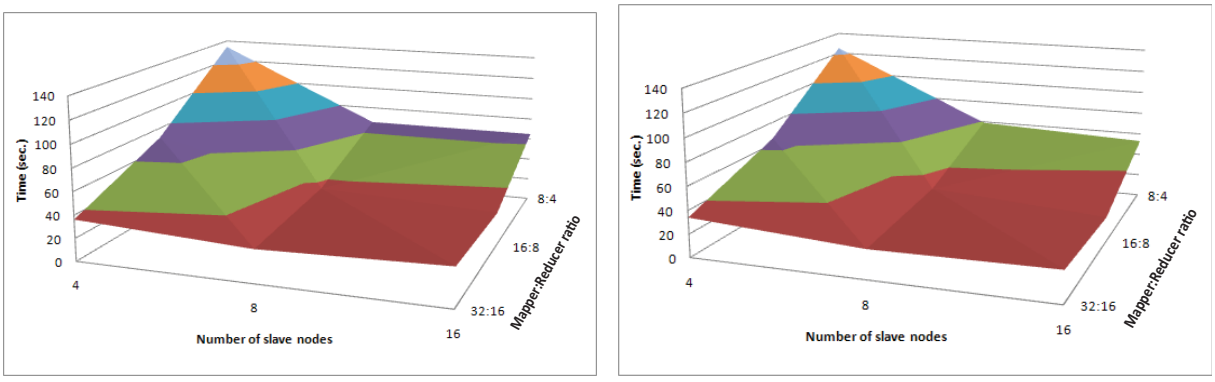
iCanCloud has been designed to obtain a good trade-off between flexibility, accuracy, performance and scalability, which makes it a powerful simulation platform for designing, testing and analyzing both actual and non-existent architectures. Although iCanCloud initially focuses on cloud computing systems without reducing accuracy in storage system by providing very high detailed models for file systems, disk drives, volume managers, block cache, block schedulers, etc. Complete high performance computing systems can be modeled using this simulation platform. The best feature of iCanCloud is its ability to model and simulate large environments (thousands of nodes) with a customizable level of detail.

Moreover, distributed applications can be simulated using this framework. In fact, in this paper we also propose a model for simulating Map-Reduce applications based on the Hadoop implementation [25].

4. Validation

In this section we present the validation process to check the accuracy of the iCanCloud framework. This process consists of the comparison between the results obtained from the real systems and the results obtained in the simulations using the iCanCloud simulation framework.

The TeraSort benchmark has been used to validate a cluster system using different configurations. This application has been executed using different parameters: size of the dataset (250MB and 1GB), the number of slaves nodes, and the ratio between the number of map-reduce processes. The architecture used in this process is a cluster that consists of 17 nodes running the Hadoop implementation of Map-Reduce. Each one contains a Intel(R) Xeon(R) CPU E5405@2GHz, with 6MB of cache, 4GB of RAM and a Seagate Barracuda 7200.11 drive. The network that interconnects the nodes is a Ethernet Gigabit

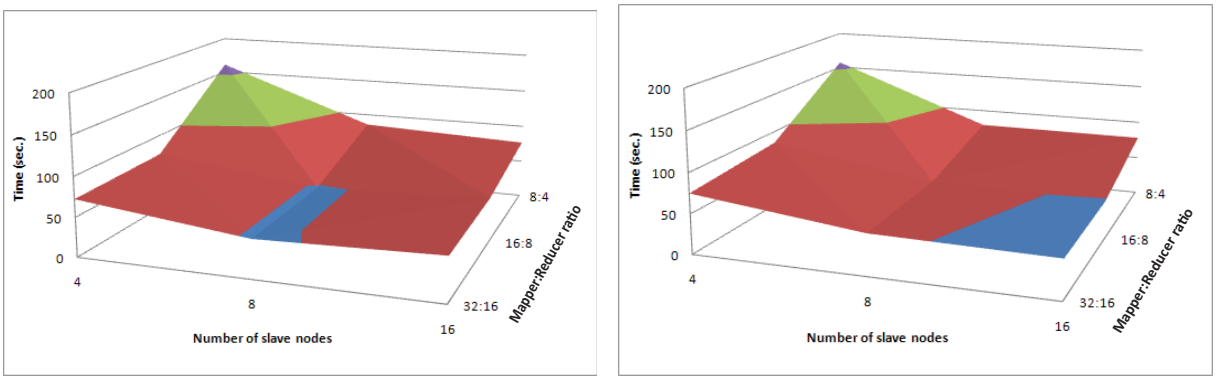


(a) Simulated execution (b) Real Model

Figure 1: Validation experiments using Terasort benchmark using a data-set of 250MB

The analysis of the obtained results is performed by the use of estimators of the accuracy and the comparison trend. The main estimator is the average accuracy of each test with different parameters, measured with a 95% confidence interval.

Figure 1 shows the results of executing the TeraSort application with a 250MB dataset. In these charts we obtain an error ratio of 8,3%. Besides this error ratio, the tendency of the simulation when different configurations are applied to the system is reflected by very high accuracy compared to the real system. Similarly, Figure 2 contains the same execution using a data-set of 1GB. In this case we obtain an error-ratio of 13,2%. However the fidelity of the simulation execution is also accurate according to the real execution.



(a) Simulated execution (b) Real model

Figure 2: Validation experiments using Terasort benchmark using a data-set of 1GB

The accuracy results presented in the graphs show the validation of the iCanCloud framework. Nearly all the accuracy results are over 90%. The most representative results involve all the subsystems of the iCanCloud framework and it is also an example of a real application. The Pearson and linear regression estimators are very near to 1 in almost all the cases. Thus, the comparison trend between real and the simulated results created a nearly straight line with a slope near to 1. The significance is that, no matter how high the absolute values are, the difference between two tests in the simulated platform will be the same as the difference between both tests in the real platform.

5. Modeling the Map-Reduce application

In this section, we propose a model for simulating the behavior of Map-Reduce using the iCanCloud simulation platform. Our proposed model has been presented with the scheme shown in Figure 3, which is a simplified version based on the Map-Reduce proposed by Google. This model uses an initial data-set as the size of the problem. By size of the problem we mean the amount of data that has to be processed in order to accomplish the execution of the application completely.

The basic idea of this model is to customize the behavior of the application by using the following set of parameters:

- Number of map processes (m): Total number of map processes.

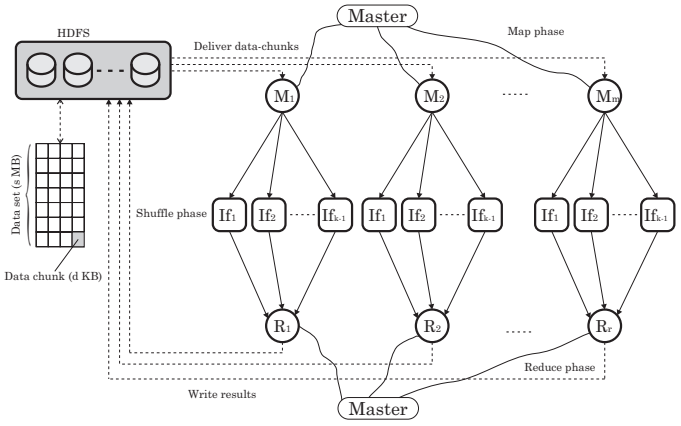


Figure 3: Map-reduce application model

- Number of reduce processes (r): Total number of reduce processes.
- Number of intermediate files (k): Number of files processed by each process.
- Size of the initial data-set (s): Size of the initial data-set (in MB), which represents the size of the problem.
- Size of data-chunk (d): Size of each domain (in KB) which is delivered to worker processes.
- Processing for each domain: Number of MIs (Million Instructions) needed for processing one domain.

6. Modeling scientific cluster architectures

With the purpose to analyze the execution of Map-Reduce model described in section 5, different multi-core scientific cluster architectures have been modeled. The main objective of those environments is to simulate the proposed Map-Reduce model using different configurations, such as the number of computing nodes, the number of CPU cores per computing node (multi-core architectures), characteristics of the network, and the number of storage nodes.

Since the Map-Reduce applications to be simulated perform data-intensive operations, different storage architectures will be deployed with the purpose of comparing the performance obtained in each. This can be done since the simulation environment where the experiments are achieved, allows for modeling a broad spectrum of storage configurations, such as centralized NFS solutions and parallel I/O architectures.

Firstly, a scenario that contains a total of 128 computing nodes has been modeled, which is shown in Figure 4. Secondly, a scenario that contains a total of 1024 computing nodes has been modeled (see Figure5).

7. Experiments and results

In this section, a set of experiments has been defined with the purpose of optimizing the architectural configuration that provides the best performance. Those experiments consist of simulating the Map-Reduce application model, as described in section 5, in the scientific computing environments described in section 6. Due to the nature of the involved application, those studies are focused mainly on the storage system, scaling both the size of the problem and the size of the environment to study the overall system performance. The size of the data-set calculated by the application is 256 GB. Basically, those experiments have been performed using different architectural parameters, like the number of CPU cores per node, the number of I/O servers and the characteristics of the communication network.

Initially, the Map-Reduce application model has been simulated using a scenario that consists of 128 computing nodes (see Figure 4). The main configuration parameters used to execute those experiments are shown in Table 1. Due to practical reasons, just the most relevant parameters are shown. In practice, a single experiment consists of hundreds

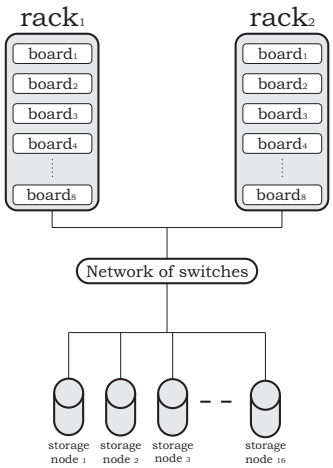


Figure 4: Science computing system scenario with 128 computing nodes

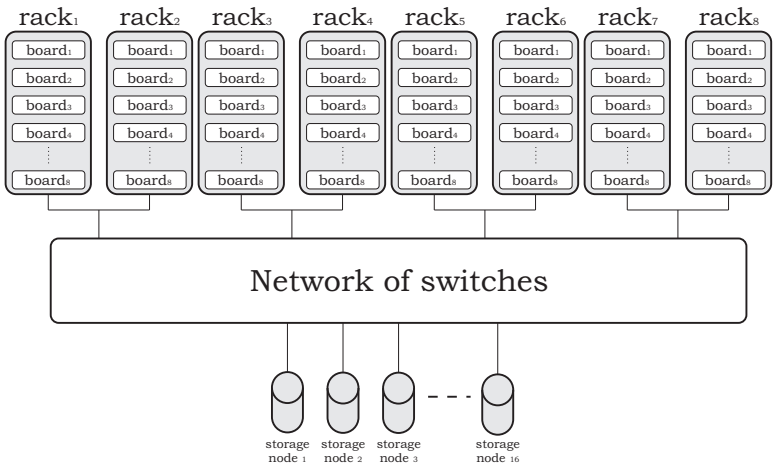


Figure 5: Science computing system scenario with 1024 computing nodes

Table 1: Map-reduce model configuration using 128 nodes

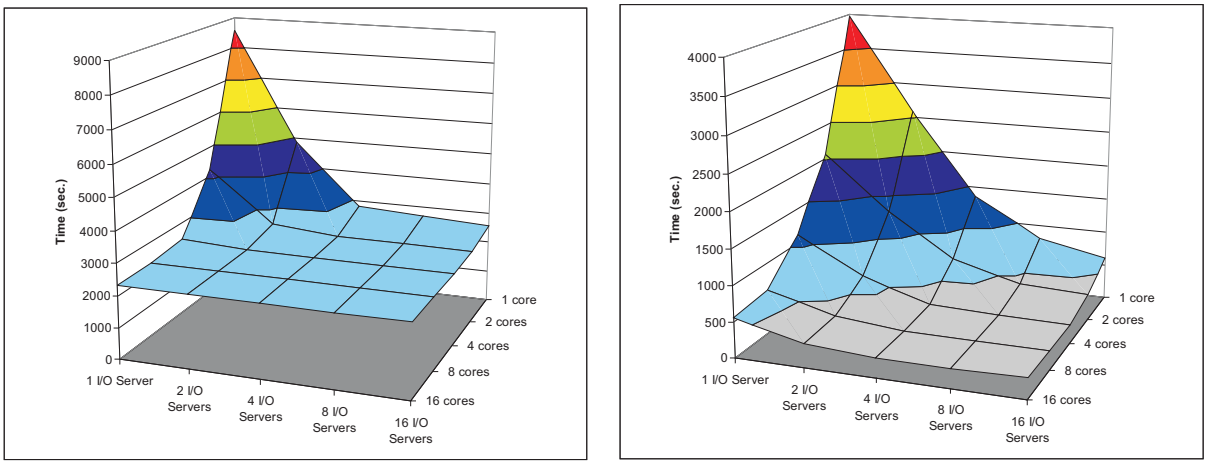
Environment model configuration	
Racks	2
Total number of computing nodes	128
CPU cores per node	1, 2, 4, 8 and 16
CPU core speed	20000 MIPS
Storage nodes	1, 2, 4, 8 and 16
File system	NFS / Parallel file system
Network	Ethernet 1 Gbps / 10 Gbps
Application model configuration	
Data-set	256 GB
Domain size	5 MB
Results size	512 KB
# processes	128, 256, 512, 1024, 2048
Processing per domain	500 MIs

of parameters. Results obtained from those simulations are grouped in two charts, depending on the communication network used. Experiments executed using a network of 1 Gbps are shown in Figure 6(a) and experiments that use a network of 10 Gbps are shown in Figure 6(b). The experiments show that there is a considerable system bottleneck when single-core CPUs and only one I/O server are used. This occurs because the level of parallelism in the system using this configuration is practically nonexistent. Otherwise, when the number of CPU cores per node or I/O servers increases, the overall system performance reflects a significant improvement.

Increasing the number of I/O servers produces a notable improvement in the overall system performance. This improvement occurs because the I/O operations of several processes are executed in parallel when several I/O servers are used. Similarly, when the number of CPU cores per node increases, the system performance also improves. However, this improvement is not reflected in all cases. For instance, in the scenario that uses a network of 1 Gbps, using more than 2 I/O servers and more than 2 CPU cores per node does not provide a significant performance improvement. Similarly, using a faster network produces a similar effect, but in this case the overall system performance remains stuck from 4 I/O servers and 4 CPU cores per node. In conclusion, using a fast network in these experiments causes a great improvement in performance for all configurations. This happens because the network acts as a system bottleneck, and increasing the network bandwidth the overall system performance increases as well.

Otherwise, increasing the number of CPU cores and I/O servers does not warrant an increase in performance since the application raises the maximum throughput provided by the system.

The next experiments are targeted to analyze the impact on the overall system performance when the number of computing nodes increases. In this case, the same application model is executed in a scenario that consists of 1024



(a) 1 Gbps network

(b) 10Gbps network

Figure 6: Simulation of the Map-Reduce model using 128 nodes

Table 2: Map-reduce model configuration using 1024 nodes

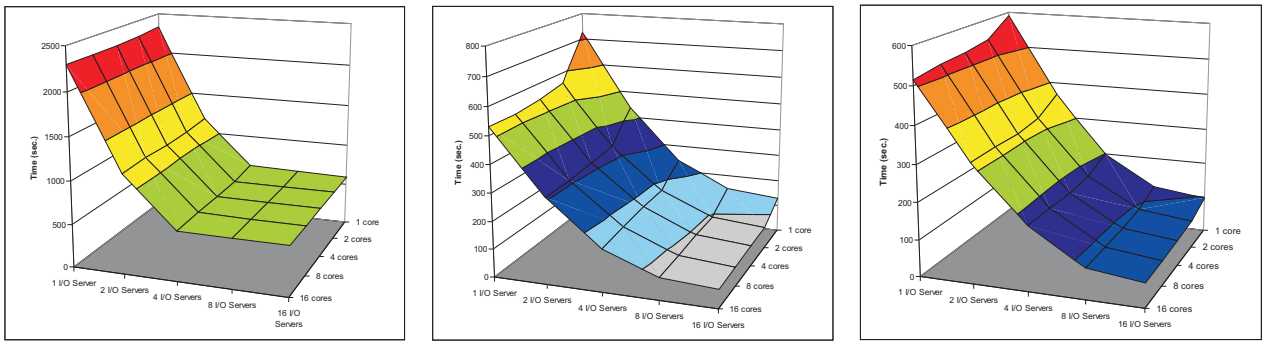
Environment model configuration	
Racks	8
Total number of computing nodes	1024
CPU cores per node	1, 2, 4, 8 and 16
CPU core speed	20000 MIPS
Storage nodes	1, 2, 4, 8 and 16
File system	NFS / Parallel file system
Network	Ethernet 1 Gbps / 10 Gbps
Application model configuration	
Data-set	256 GB
Domain size	5 MB
Results size	512 KB
# Processes	1024, 2048, 4096, 8192 and 16384
Processing per domain	500 MIs

computing nodes (see Figure 5) using the configuration shown in Table 2.

Figures 7(a), 7(b) and 7(c) show the results obtained from those simulations. Initially, Figure 7(a) shows the results of the experiments using a network of 1 Gbps. Next, Figure 7(b) shows the results of the experiments using two different networks, a 1 Gbps network that interconnects the racks with the network of switches (network for communications) and another network of 10 Gbps that interconnects the network of switches with the storage servers (I/O system network), Finally, Figure 7(c) shows the results of the experiment using a network of 10 Gbps.

These charts clearly show that using a different number of CPU cores per node in this scenario doesn't give a significant improvement in the overall system performance. Only when a network of 10 Gbps is used in the I/O system, dual-core CPUs provide a small increase in performance. In the remaining configurations, the overall system performance does not change. Otherwise, increasing the number of I/O servers produces a significant improvement in performance. This happens because by using 1024 computing nodes, the bottleneck of the system lays more aggressively in both the network and the storage systems. Therefore, increasing the number of I/O servers, the level of parallelism in the storage system increases as well, alleviating the system bottleneck. In the first scenario (see Figure 7(a)) the overall system performance is stuck from 4 I/O servers. Otherwise, the other scenarios (see figures 7(b) and 7(c)) present an increase in performance up to 8 I/O servers.

Comparing the experiments that use a mixed network (1 Gbps for communications and 10 Gbps for the storage subsystem) with those experiments that use a slow network (1 Gbps), we can conclude that most of the bottlenecks lie



(a) 1 Gbps network (b) 1Gbps/10Gbps network (c) 10 Gbps network

Figure 7: Simulation of the Map-Reduce model using 1024 nodes

in the storage system, since using the same bandwidth for communications and a faster network for the I/O subsystem, the overall system performance clearly improves.

In conclusion, these experiments show that the I/O system is a critical factor in HPC systems, obtaining a greater performance by increasing the network bandwidth instead of the number of CPUs per node or the number of I/O servers. When a 10 Gbps network is used in the I/O system, the overall system performance reflects a great improvement, which is not a noticeable fact when a 1 Gbps network is used (see Figure 7(c)).

This architecture does not provide a significant improvement in performance when more CPU cores are used.

Using a faster network has different effects on the tests, depending on the number of I/O servers used. When the experiment uses only one I/O server, the overall system performance is practically the same as when it uses both networks. Otherwise, when several I/O servers are used, the 10 Gbps network provides a greater performance.

Finally, increasing the number of I/O servers provides a remarkable increase in performance. This can be appreciated using both networks. However, the difference in performance is greater in those configurations that use a 10 Gbps network. Therefore, the overall system performance in this scenario is improved when the number of I/O nodes increases, since the level of parallelism in the I/O system increases as well.

8. Conclusions and future work

In this paper we have performed a study of the compared performance of scaling up scientific multi-core cluster architectures using a Map-Reduce application model. This evaluation has been performed by using a simulation framework, called iCanCloud, specifically using its underlying properties for modeling and simulating scientific computing architectures.

The results obtained show that increasing the number of cores does not have a significant impact on the performance when it is not followed by the equivalent rise of the communication bandwidth and the storage capabilities. In fact, the bandwidth of the communication between the computing nodes and the storage servers is the most critical bottleneck, which can hinder not only the improvement on the computing cores, but also the improvement on the storage servers.

The simulated Map-Reduce application presents a great increase in performance in the three previous architectures when a fast network is used. This improvement in performance happens mainly because the network acts as a system bottleneck, specifically in the storage system. That indicates the importance of scaling the network the same as the rest. Increasing both the number of I/O servers and CPU cores per node provides an increase in performance in some cases, which depends both on the architecture and on the configuration used. Generally, this performance remains fixed when the application raises the maximum throughput provided by the system configuration. This problem is emphasized on many-core nodes. The network interface shared by all the cores causes a significant bottleneck in the system.

The experiments have demonstrated that increasing the number of resources, such as the number of CPU cores per node, does not always warranty a significant improvement in the overall system performance whether the rest of

the resources are not scaled up adequately. Thus, using the optimal architecture configuration can obtain the same performance at less cost than other architectures with more resources.

Future work may include modeling and simulating other application types that are usually executed in distributed and parallel systems. Moreover, other architectures that use several network interfaces per computing node will be simulated with the purpose of alleviating the bottleneck focused in the network interface.

Acknowledgment

This research was partially supported by the Spanish Ministry of Science and Innovation under the grants TIN2010-16497 and TIN2009-14312-C02-01 and the Santander-UCM Programme to fund research groups (GR35/10-A - group number 910606).

References

[1] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, Communications of the ACM 51 (1) (2008) 107–113.
[2] S. Sehrish, J. Bent, J. Lopez, S. Habib, J. Wang, Introducing Map-Reduce to high end computing, in: PDSW'08: 3rd Petascale Data Storage Workshop, 2008.
[3] A. Matsunaga, M. Tsugawa, J. Fortes, Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications, in: eScience'08: IEEE Fourth International Conference on eScience, 2008, pp. 222–229.
[4] H. Meuer, E. Strohmaier, J. Dongarra, H. D. Simon, Top500 Supercomputer sites, http://www.top500.org (2010).
[5] A. Nunez, G. Castane, J. Vazquez-Poletti, A. Caminero, J. Carretero, I. Llorente, Design of a flexible and scalable hypervisor module for simulating cloud computing environments, in: Performance Evaluation of Computer Telecommunication Systems (SPECTS), 2011 International Symposium on, 2011, pp. 265 –270.
[6] E. Argollo, A. Falcón, P. Faraboschi, M. Monchiero, D. Ortega, COTSon: infrastructure for full system simulation, SIGOPS Operating Systems Review 43 (1) (2009) 52–61.
[7] M. Rosenblum, S. A. Herrod, E. Witchel, A. Gupta, Complete computer system simulation: the SimOS approach, Parallel & Distributed Technology: Systems & Applications, IEEE 3 (4) (1995) 34–43.
[8] M. Rosenblum, E. Bugnion, S. Devine, S. A. Herrod, Using the SimOS machine simulator to study complex computer systems, ACM Transactions on Modeling and Computer Simulation 7 (1) (1997) 78–103.
[9] A. F. Rodrigues, R. C. Murphy, P. Kogge, K. D. Underwood, The structural simulation toolkit: exploring novel architectures, in: SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing, ACM, New York, NY, USA, 2006.
[10] R. Buyya, M. Murshed, GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing, Concurrency & Computation: Prac. & Exp. 14 (2002) 1175–1220.
[11] W. H. Bell, D. G. Cameron, L. Capozza, A. P. Millar, K. Stockinger, F. Zini, Simulation of dynamic grid replication strategies in OptorSim, in: Proc. of the 3rd Intl. Workshop on Grid Computing (Grid), Baltimore, USA, 2002.
[12] K. Fujiwara, H. Casanova, Speed and accuracy of network simulation in the simgrid framework, in: Proc. of the 1st Intl. Workshop on Network Simulation Tools (NSTools), Nantes, France, 2007.
[13] X. Liu, Scalable Online Simulation for Modeling Grid Dynamics, Ph.D. thesis, Univ. of California at San Diego (2004).
[14] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, R. Buyya, CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software: Practice and Experience 41 (1) (2011) 23–50.
[15] S.-H. Lim, B. Sharma, G. Nam, E.-K. Kim, C. R. Das, MDCSim: A multi-tier data center simulation, platform, in: Intl. Conference on Cluster Computing and Workshops (CLUSTER), New Orleans, USA, 2009.
[16] L. Adhianto, S. Banerjee, M. Fagan, M. Krentel, G. Marin, J. Mellor-Crummey, N. R. Tallent, HPCToolkit: tools for performance analysis of optimized parallel programs, Concurrency and Computation: Practice and Experience 22 (6) (2010) 685–701.
[17] M. Geimer, F. Wolf, B. J. N. Wylie, E. Ábrahám, D. Becker, B. Mohr, The Scalasca performance toolset architecture, Concurrency and Computation: Practice and Experience 22 (6) (2010) 702–719.
[18] B. Penoff, A. Wagner, M. Tüxen, I. Rüngeler, MPI-NeTSim: A Network Simulation Module for MPI, in: ICPADS'09: IEEE 15th International Conference on Parallel and Distributed Systems, IEEE, 2009, pp. 464–471.
[19] T. Sun, C. Shu, F. Li, H. Yu, L. Ma, Y. Fang, An efficient hierarchical clustering method for large datasets with map-reduce, in: PDCAT '09: International Conference on Parallel and Distributed Computing, Applications and Technologies, IEEE Computer Society, Washington, DC, USA, 2009, pp. 494–499.
[20] K. Wang, J. Han, B. Tu, J. Dai, W. Zhou, X. Song, Accelerating spatial data processing with mapreduce, in: ICPADS'10: IEEE 16th International Conference on Parallel and Distributed Systems, 2010, pp. 229–236.
[21] I. Palit, C. Reddy, Parallelized boosting with map-reduce, in: ICDMW '10: IEEE International Conference on Data Mining Workshops, 2010, pp. 1346–1353.
[22] M. Elteir, H. Lin, W. chun Feng, Enhancing mapreduce via asynchronous data processing, in: ICPADS'10: IEEE 16th International Conference on Parallel and Distributed Systems, 2010, pp. 397 –405.
[23] M. Deodhar, C. Jones, J. Ghosh, Parallel simultaneous co-clustering and learning with map-reduce, in: GrC'10: IEEE International Conference on Granular Computing, 2010, pp. 149–154.
[24] A. Núñez, , G. Castañé, J. Carretero, The iCanCloud simulation platform, http://icancloudsim.org (2011).
[25] Apache hadoop, http://hadoop.apache.org/ (2012).